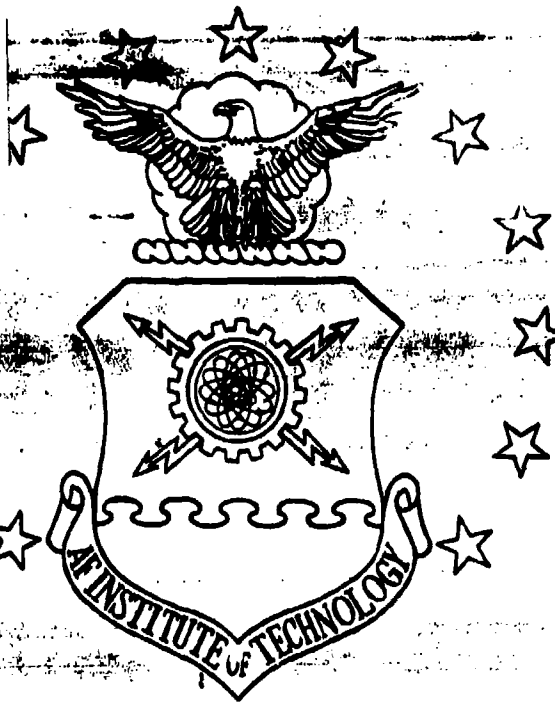


DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/91D-49

INFRARED TARGET RECOGNITION

THESIS

Brian D. Singstock
Second Lieutenant, USAF

AFIT/GE/ENG/91D-49

AFIT/GE/ENG/91D-49

INFRARED TARGET RECOGNITION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Brian D. Singstock, B.S.E.E.
Second Lieutenant, USAF

December, 1991

Acknowledgments

I would like to thank my advisor Major Steve Rogers for his invaluable direction and support during this escapade. Unfortunately, the path I started out on ran short of where I needed to go, and Major Rogers broadened my vision enough to forsee a new path to take. Fortunately, this one did lead me to the end.

I would also like to thank Captain's Pedro Suarez and Jim Goble for the donation of their Karhunen–Loève Transform and Discrete Cosine Transform code. Without it I could not have finished my thesis on time.

I would like to thank Captain Ken Fielding for the VHS tape of excellent FLIR images he provided.

Finally, I would like to extend a little more than the customary expression of appreciation to my two favorite women: Leslie and Treukka, my 6 month old yellow lab. Without the both of you to provide the much needed distractions, the days would have been much longer.

Brian D. Singstock

Table of Contents

List of Figures

Figure Page

List of Tables

Abstract

In this thesis, three approaches were used for Automatic Target Recognition (ATR). These approaches were shape, moment and Fourier generated features, Karhunen–Loève Transform (KLT) generated features and Discrete Cosine Transform (DCT) generated features. The KLT approach was modelled after the face recognition research by Suarez, AFIT, and Turk and Pentland, MIT. A KLT is taken of a reduced covariance matrix, composed of all three classes of targets, and the resulting 'eigenimages' are used to reconstruct the original images. The reconstruction coefficients for each original image are found by taking the dot product of the original image with each 'eigenimage'. These reconstruction coefficients were implemented as features into a three layer backprop with momentum network. Using the hold–one–out technique of testing data, the net could correctly differentiate the targets 100% of the time. Using standard features, the correct classification rate was 99.33%. The DCT was also taken of each image, and 16 low frequency Fourier components were kept as features. These recognition rates were compared to FFT results where each set contained the top five features, as determined by a saliency test. The results proved that the DCT and the FFT were equivalent concerning classification of targets.

INFRARED TARGET RECOGNITION

I. INTRODUCTION

1.1 Background

The military has presented a need for an automatic target recognizer (ATR) which would remove the dependency on a human interface to aim a missile in a battlefield scenario. The missile would be sent to search a certain area for potential targets and take out the most important of the targets found. Most of the 'searching' has been done with infrared sensors, however, Laser RADAR (LADAR), Synthetic Aperature RADAR (SAR) and Millimeter Wave (MMW) are three other sensors also being tested. With the infrared images, all the approaches to ATR deal with standard feature extraction methods, like finding the length–to–width ratio of the segmented target. However, alternate approaches have been tried in the realm of face recognition. Two such approaches deal with Karhunen–Loève Transforms (KLT) and Discrete Cosine Transforms (DCT). The eigenimages (eigenvectors) from a Karhunen–Loève transform could be used to reconstruct the original images, and the reconstruction coefficients could be used as features for target recognition. This approach has been done for face recognition producing very positive results (18, 15). Also, Discrete Cosine Transforms have been used in face recognition research replacing the Fast Fouier Transform (FFT) for the generation of the low frequency Fourier components (7).

1.2 Problem Statement

Very little work has been done beyond the standard feature extraction methods in target recognition. We will compare the standard feature extraction technique to two new techniques. One of these will find 'eigenimages' based on the eigenvectors produced from the KLT of a reduced covariance matrix. Then, the eigenimages will be used to reconstruct

the original images. The reconstruction coefficients will be implemented as features to a neural network which will classify the targets. The second new technique will use a DCT to generate Fourier components. The low frequency components will be implemented as fetaures to the same neural net which will be used for classification.

1.3 Research Objectives

This thesis will compare the use of a KLT to produce 'reconstruction features', and standard features in training and testing a neural network for ATR. Also, it will compare the use of DCT generated features in target recognition to FFT generated features.

1.4 Research Questions

1. Will a KL image reconstruction feature set, using a reduced covariance matrix, be able to recognize targets?
2. Will the KL image reconstructed feature set work better than a standard feature set?
3. Will a DCT be able to recognize targets?
4. Will DCT features work better than standard FFT and/or shape features in target recognition?

1.5 Approach

1.5.1 **Target Segmentation** 512x512 images will be captured from a VHS tape of FLIR data. Each image contains three classes of targets: tanks, jeeps and towers. The target and local background will be cropped out from the original image, histogram equalized, and thresholded. The thresholded and cropped images will be reduced to 128x128 pixels, and the thresholded, binarized images will be used as 'segmented' targets.

1.5.2 **Standard Feature Generation** Intensity, shape and Fourier feature sets will be generated and tested independently as well as together for target recognition. The shape features will be generated using the segmented images.

1.5.3 **KL Transform** A reduced covariance matrix will be found using a subset of the entire image set. Eigenimages and eigenvalues will be generated from this reduced covariance matrix, and the best eigenimages will be kept based on a percentage of the eigenvalue's energy desired to keep for reconstruction, $\approx 99\%$. All of the original images will be reconstructed using this reduced eigenimage set, and the reconstruction coefficients for each eigenimage will be used as features in a neural network.

1.5.4 **Discrete Cosine Transform** A Discrete Cosine Transform (DCT) will be computed for each image, and a small portion of the low frequency Fourier components will be kept as features. These results will be compared to those found using an FFT to generate the features.

1.5.5 **Hardware** All the feature sets will be tested on a three layer backprop with momentum neural network. The hidden layer will consist of 7 nodes. All the results will be determined using the hold–one–out technique of testing (14). Saliency will also be used to reduce the feature sets so that Foley's criteria, explained later, will be met.

II. LITERATURE REVIEW

2.1 Introduction

The military would like an automatic target recognizer to put on the front end of its various munitions. The advantage of such a system is in the minimalization of friendly life loss due to the removal of any human interface in the targeting aspect. The immediate approach was to model the human visual system (HVS). However, no one is sure how the HVS works or the interaction of the human mind in adjusting the picture the eye sees. It appears that the HVS can adjust in most cases for scale, rotation and aspect of an image. It cannot memorize pictures it has not seen, yet a new angle of viewing a hammer tells us it is still a hammer. While the HVS works great, the military is still stuck. Maybe, we, as engineers, do not need to know how computers recognize targets, but only that they can–the same idea as the HVS. This leads to the approach of giving a computer an entire image and letting it decide what is and is not important in determining the presence of a target. This idea would currently be too computationally intensive due to the number of pixels per image required for decent resolution; however, different approaches are available to reduce the size of data given to the computer. These different approaches involve generating a set of features that can linearly separate the different target classes from each other. Unfortunately, no one knows what measurements and computations comprise the perfect feature set, instead many approaches are tried.

2.2 Neural Networks

A brief explanation of neural networks will be included since a neural net will be used for all feature set processing. Hopefully this explanation will explain why the feature sets are being generated. The basic idea behind a neural net is that the computer is given a feature set for each target, an exemplar, and it is told to derive a known output, the target type. If the net works on the first try, it is left alone. If it doesn't, the weights which produce the output are adjusted towards producing the correct output. The next exemplar is processed,

and an output is produced for it. Again the weights are updated depending on the correct classification of the output (12, 13, 14). Eventually, the computer 'learns' how to produce the desired output for a given input. One advantage to this process is that the user does not need to know how the network works, only that it does, like the HVS.

The feature sets that are given to a neural net to train and test with need to be generated in some fashion. Three different sets include standard features, KLT reconstruction features and DCT low frequency Fourier features. Each is discussed below.

2.3 Standard Feature Set Generation

Standard feature sets include Fourier components, intensity values and shape features. The Fourier components are typically low frequency components, and the intensity features typically involve ratios between different local areas, such as target/non-target. The shape features need some type of truthing image of the original image to compute them. These values include moments, length to width ratios, complexity, etc. However, the need to know which pixels are target pixels is necessary.

2.3.1 **Moment Features** Moments can be used to reconstruct the images they were generated from if enough are included (17). This is the motivation to include moments as features. However, up through third order moments are usually the minimal amount to classify targets. This turns out to be quite a few features, ten to be exact. Michael Teague's article on moments mentioned that first and second order moments together completely specify an ellipse (17).

Since an ellipse only needs its semi-major, a, and semi-minor, b, axes to be regenerated, these two values could replace the moments through second order which turns out to be a reduction of six features to two.

$$M_{i,j} = \iint f(x,y)\, x^i\, y^j\, dx\, dy \tag{1}$$

$$a = \sqrt{\frac{M_{20} + M_{02} + [(M_{20} - M_{02})^2 + 4M_{11}^2]^{1/2}}{M_{00}/2}} \quad (2)$$

$$b = \sqrt{\frac{M_{20} + M_{02} - [(M_{20} - M_{02})^2 + 4M_{11}^2]^{1/2}}{M_{00}/2}} \quad (3)$$

Equation 1 is used to find each moment, and Equations 2 and 3 are used to find the semi-major axis, a, and semi-minor axis, b.

Each of the moments were normalized using target area squared, Equation 4.

$$normal = (area)^2 \quad (4)$$

There is no particular justification to choosing this normalization value except it is related to the size of the target. Inside of Equation 1, the kernel $f(x, y)$ describes the brightness of the target at (x, y).

By varying the density function of the target, different moments can be calculated. Three of these are listed below:

silhouette moment making the kernel one for a target pixel and zero otherwise,

outline moment making the kernel one for a target outline pixel and zero otherwise, and

standard moment making the kernel equal to the gray scale value for target pixels and zero otherwise.

When using the standard moment approach, a problem usually arises concerning the magnitude of the moment, it is oftentimes too large for a program variable to store. Because of this standard moments are usually replaced with either silhouette or outline moments.

2.3.2 **Shape and Intensity Features** Besides moment features, there are other shape features that are typically used. Some of these are

length–to–width ratio computed for a rectangle barely encompassing the entire target,

complexity ratio of the # of perimeter pixels to the area of the target,

compactness ratio of the area of the target to the area of a rectangle barely encompassing the entire target,

perimeter^2/area self-explanatory, and

boxness ratio of target perimeter to perimeter of rectangle from above.

Shape features, such as the ones listed above, have been the bread and butter of many previous feature sets used in target recognition. About the only other type of feature used previously is related to intensity.

Temperature intensity features can only be used with infrared images. They relate the temperature in one section of the image to the temperature in another. Some common intensity measurements are:

(max intensity - min intensity)/average intensity self-explanatory,

max intensity to average intensity ratio self-explanatory,

max intensity self-explanatory, and

gray level contrast ratio of the average target intensity to the average background intensity.

In a multisensor environment, this is where FLIR imagery would contribute the most—it adds features not measurable by other sensors.

2.3.3 **Low Frequency Fourier Features** Low frequency Fourier components can reproduce a very close approximation of an image with an inverse FFT. Everything but the sharp changes which require higher frequency components can be restored. Based on this reasoning, Fourier components have been used as features for target recognition.

2.4 Feature Set Generation Using the Karhunen–Loève Transform of a Coverience Matrix

A KL transform of the covariance matrix of a set of images creates a new orthogonal basis set for those images based on the statistical properties of them. The importance of each new orthogonal dimension, described by its eigenimage, can be judged by the size of its eigenvalue. By keeping only the most important eigenimages, the amount of reconstruction of the original image can be determined by an energy ratio of the eigenvalues from the eigenimages used for recognition to the total eigenvalues. The original images can be reconstructed using only the eigenimages, and these reconstruction coefficients can be used as features in a neural network. If the entire covariance matrix were used, each image could be reconstructed exactly using the eigenimages; however, if a reduced set of the eigenimages were used, the best representation of the original image by a reconstructed image would be based on a minimization of a Mean Squared Error term. The fact that the MSE term is always minimized for the number of eigenimages used, ie. the dimensionality of the new feature space, is the advantage of using a KLT on the covariance matrix.

2.4.1 **Normal KL Transform Mathematical Derivation** The eigenvalues and eigenimages are determined for the covariance matrix of the input images. First, each NxN image is stored in a N^2 vector:

$$\mathbf{x}_i = \begin{bmatrix} x_i[1] \\ x_i[2] \\ \vdots \\ x_i[j] \\ \vdots \\ x_i[N^2] \end{bmatrix} \tag{5}$$

where $x_i[j]$ corresponds to the j^{th} pixel value of the i^{th} image. The covariance matrix is defined as:

$$\mathbf{C}_\mathbf{x} = E\left\{(\mathbf{x} - \mathbf{m}_\mathbf{x})(\mathbf{x} - \mathbf{m}_\mathbf{x})^T\right\} \tag{6}$$

where

$$\mathbf{m_x} = E\{\mathbf{x}\} \tag{7}$$

According to Gonzalez and Wintz (8), the mean vector and covariance matrix can be approximated with the following:

$$\mathbf{m_x} \cong \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i \tag{8}$$

where M is the number of NxN images, and

$$\begin{aligned}\mathbf{C_x} &\cong \frac{1}{M}\sum_{i=1}^{M}(\mathbf{x}_i - \mathbf{m_x})(\mathbf{x}_i - \mathbf{m_x})^T \\ \mathbf{C_x} &\cong \frac{1}{M}\left[\sum_{i=1}^{M}\mathbf{x}_i\mathbf{x}_i^T\right] - \mathbf{m_x}\mathbf{m_x}^T\end{aligned} \tag{9}$$

The covariance matrix will have N^2 eigenimages, $\mathbf{e}_i$, and corresponding eigenvalues, λ_i. These N^2 eigenimages can be arranged in a N^2xN^2 matrix, $\mathbf{A}$, where the eigenimages have been arranged in decreasing order as determined by the magnitude of their respective eigenvalues, λ_i.

$$\mathbf{A} = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1N^2} \\ e_{21} & e_{22} & \cdots & e_{2N^2} \\ \vdots & \vdots & \vdots & \vdots \\ e_{N^21} & e_{N^22} & \cdots & e_{N^2N^2} \end{bmatrix} \tag{10}$$

where e_{ij} is the j^{th} component of the i^{th} eigenimage. Now let

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{m_x}) \tag{11}$$

and define the covariance matrix of $\mathbf{y}$, C_y, the same way it was defined for $\mathbf{x}$ in Equation 6. It can be shown that:

$$\mathbf{C_y} = \mathbf{A}\mathbf{C_x}\mathbf{A}^T \tag{12}$$

where C_y is a diagonal matrix with its values equal to the ordered eigenvalues of the original covariance matrix, C_x.

$$\mathbf{C_y} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & \dots & \lambda_j & \dots \\ 0 & 0 & \dots & \lambda_{N^2} \end{bmatrix} \tag{13}$$

Because the covariance matrix above has values only on the diagonal, the eigenvectors of the matrix are orthogonal, changing one value will have no affect on any other value. This proves the orthogonality of the KL transform, and makes the KL transform an excellent method of reducing dimensionality, including in feature sets, because it finds the best representation of feature set.

2.4.2 **Reduced Covariance Matrix KL Transform** Instead of using the entire N^2xN^2 covariance matrix, a reduced covariance matrix can be found. This approach is modelled in Turk and Pentland where it was used for face recognition (18, 15). According to Turk and Pentland,

> If the number of data points in the image space is less than the dimension of the space ($M < N^2$), there will be only $M - 1$, rather than N^2, meaningful eigenvectors. (The remaining eigenvectors will have associated eigenvalues of zero.) (18:6)

The reduction is from an N^2xN^2 matrix to a MxM matrix, where N is the number of rows/columns in the image and M is the number of images used in the KL transform. Assuming that 10 images are used to produce the reduced covariance matrix, the reduction in the number of elements for a 128x128 image is $2.68x10^8$ to 100, a factor of $\approx 10^6$. This is quite a difference. Using the reduced covariance matrix, the all the images will not be completely restorable; however, the question of importance to this research is whether the reconstruction coefficients will be able to separate the target classes for classification of an independent test set.

The reduced covariance matrix was found by setting up an N^2xM matrix:

$$\mathbf{A} = \begin{bmatrix} \mathbf{x}_1[1] & \mathbf{x}_2[1] & \dots & \mathbf{x}_M[1] \\ \mathbf{x}_1[2] & \mathbf{x}_2[2] & \dots & \mathbf{x}_M[2] \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_1[N^2] & \mathbf{x}_2[N^2] & \dots & \mathbf{x}_M[N^2] \end{bmatrix} \tag{14}$$

where $\mathbf{x}_i[j]$ is the j^{th} pixel value of the i^{th} training image. The reduced covariance matrix **L** is found by multiplying the **A** matrix by its transpose:

$$\mathbf{L} = \mathbf{A}^T\mathbf{A} \tag{15}$$

This produces an MxM matrix which is used as the new covariance matrix. The eigenimages and eigenvalues can be found now. There will be M of each. More details involving this reduced covariance matrix can be found in Suarez's thesis and Turk and Pentland's research (15, 18).

2.5 Feature Set Generation Using a Discrete Cosine Transform

The DCT is a modified FFT where the original image is entirely real and symmetric so that the transform would also consist of only real values. The immediate advantage is in the computation. The phase terms turn out to be only real values so that no complex math is required with this transform. This makes the DCT extremely fast. The DCT produces a 128x128 transformed image of an original 128x128 image. The only quadrant from the FFT that is produced is the first, the one containing only positive frequencies.

The DCT is a simplified Discrete Fourier Transform (DFT). It can be derived from the DFT equation if the image is real and it has even symmetry, see Figure 1. However, the image need not demonstrate even symmetry for the DCT to work, see Appendix A. The math involved in this relationship is rather painstaking so it has been deferred to an appendix.

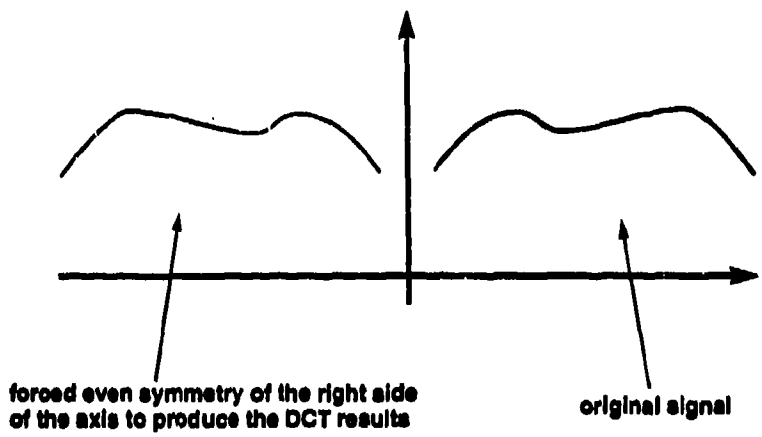


Figure 1. The forced even symmetry necessary to derive the DCT from the FFT.

2.5.1 **DCT Mathematics** The DCT kernel can be derived directly from the FFT kernel by taking the real part of the FFT equation (8). In one dimension, this is shown below:

$$C(u) = \sqrt{\frac{2}{N}} Re\left\{\left[exp(\frac{-j2\pi}{2N})\right] \sum_{x=0}^{2N-1} f(x) exp(\frac{-j2\pi ux}{N})\right\} \tag{16}$$

$$u = 1, 2, \ldots, N-1$$

$$C(u) = \sqrt{\frac{2}{N}} \sum_{x=0}^{N-1} f(x) \cos\frac{(2x+1)u\pi}{2N} \tag{17}$$

The above derivation is based on Euler's equation:

$$e^{jx} = \cos x + j \sin x \tag{18}$$

By taking the real part of the above equation, the cosine term is the only remaining term. The two dimensional DCT equation is listed below:

$$C(0,0) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \tag{19}$$

$$C(u,v) = \frac{1}{2N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \left[\cos\frac{(2x+1)u\pi}{2N}\right] \left[cos\frac{(2y+1)v\pi}{2N}\right] \tag{20}$$

for u, v = 1,2,...,N-1.

2.5.2 **Implementation** The DCT based system is implemented much as any Fourier coefficient recognizer would be (7). The low frequency components are used as feature inputs to a neural net. The neural net trains off of these values.

2.6 Feature Set Generation Using a Cottrell Network

Garrison Cottrell has developed a backprop neural network that can generate its own non-specific features for face recognition (2, 4).

> The model is accurate at distinguishing faces from non-faces, recognizing new instances of familiar faces, and determining the faceness and, to a degree, sex of new faces (4:1).

Cottrell's network may be general enough to expand it to all areas of pattern recognition, including speech and target recognition. Choosing good features is a field in itself concerning ATR, and the use of a Cottrell network to decide which features it wants to use may be a better approach than having a human determine 'optimal' specific features.

Cottrell has shown that a 64x64 image can be regenerated by only storing 40 features of the image, called 'holons'. This is a reduction of 4096 inputs by a factor of over 100, see Figure 2. By properly training the network towards being a target classifier, the 'auto-associative' level of Figure 2 could generate a number of non-specific features for input to the 'face information extractor' which might be termed 'target classifier'. This target classifier would look for targets instead of gender and emotion.

2.7 Feature Set Reduction

There are a few rules that need to be adhered to if the results generated are to be valid. These rules are listed below:

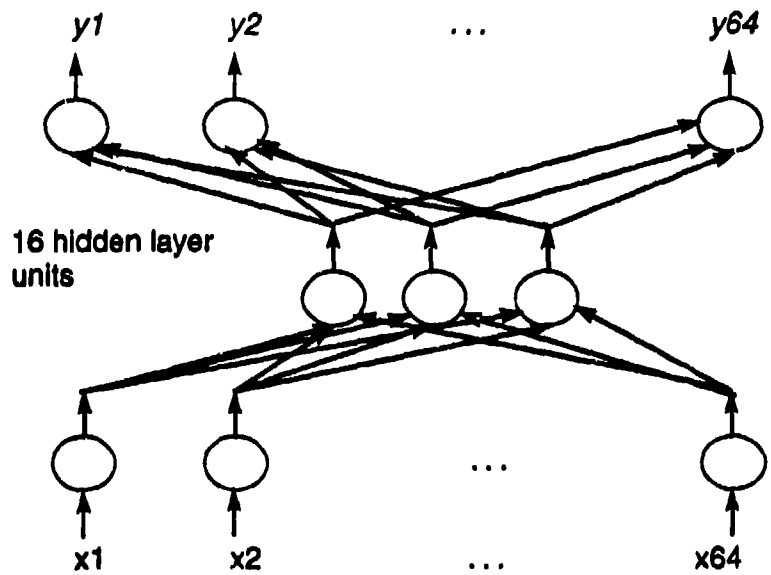


Figure 2. A Cottrell network.
(2:2)

Foley's rule for each class of outputs you have, you need 3 times the number of input features for your training vectors (5),

Cover's rule if you have less than 2 times the number of input features per class of outputs for training vectors, the classifier has a 50% probability of being 100% trainable because the problem has a 50% probability of being linearly separable (3), and

'Uncle Bernie's' rule the number of training vectors must be greater than or equal to W/ϵ, where W is the number of weights and ϵ is the error percentage desired. This rule came from personal communication with Bernie Widrow.

Now it isn't always easy to generate more exemplars so often the feature set is reduced instead. There are a few approaches that can be used to reduce the feature set: Karhunen-Loève transform, Cottrell net and saliency. Each is explained below.

2.7.1 **Karhunen–Loève Transform** The Karhunen-Loève transform is the same that was described above. The KLT finds a best representation of all the features in an

orthogonal feature space. The eigenvectors will be orthogonal to each other and will be linear combinations of the original features. By examining the eigenvalues associated with each eigenvector and ignoring the small ones, the set of eigenvectors can be reduced. This reduces the number of features to the neural network while still using all the original features. The weights between the original features and the new 'reduced feature set' are fixed so the transform is extremely fast.

2.7.2 **Cottrell Network** A Cottrell net can be used any time there is a need to reduce the number of elements in a set. The Cottrell net finds an identity matrix that maps the original feature space to the new reduced feature space, where the size of the new set must already be specified, and back to the original feature space. The new reduced feature set can be sent to a neural net for processing without the need to regenerate the original feature set. This again is a summation of fixed calculations between the two sets; however, the weights are determined by training the Cottrell net to find an approximation of the identity matrix.

2.7.3 **Saliency** Saliency is a measure of the 'goodness' of a certain feature with respect to all the other features in the feature set (14). The above two approaches still use the entire original feature set as input to the feature set reducers so they do not eliminate the useless ones. Saliency can eliminate the useless features from a set. The basic idea is that an important feature would have a large weight associated with it so that it could affect the training of the net more than an unimportant feature which would have a small weight associated with it. By taking only those features with large weights associated with them, you select only the most salient features of the set. The problem with this approach is that of relativity. This method, as with any other, only tells the user the importance of one feature relative to a certain set of features. It does not predict how a feature will do in another set of features or by itself.

Using the weight values as a test of saliency between features has been proven to be a very close approximation to actually calculating the contribution of a given feature to

the output (16). So by examining the weights, the saliency routine is much faster. The only catch with this approach is the need to statistically normalize the data prior to any processing.

2.8 Summary

The need for an ATR system has been presented. Differing approaches have been tried; however, none has incorporated the use of eigenimages outside of face recognition (15, 18). Maybe the approach that worked so well for face recognition could also be used in generic target recognition. Also, the use of a DCT verses an FFT has yet to be compared to determine the advantages, if any, in ATR systems. The next chapter will explain the methodology behind the research as well as the procedures used.

III. METHODOLOGY

3.1 Introduction

All the images used in this research were infrared. These images included jeeps, tanks and C^4I towers. The images were cropped and 'segmented' on **KHOROS**, a data manipulation package from the University of New Mexico. Using the segmented images as truth images, basic features for each image were computed and then processed on **NEURAL GRAPHICS**(16). Next a Karhunen-Loève transform of a reduced covariance matrix for a subset of the images was taken and general eigenimages were found. These eigenimages were used to reconstruct each original image, and the resulting reconstruction coefficients were processed again on **NEURAL GRAPHICS** as features. Finally, a DCT of the images was computed, and the low frequency components were used as features. These results were then compared to the previous work done with an FFT in the 'basic features' area mentioned above.

3.2 Target Segmentation

The FLIR images were originally stored on VHS tape. The desired images were converted to frames, moved onto a SUN workstation, and preprocessed there using **KHOROS**. The images were originally 512x512 pixels but were eventually reduced to 128x128 pixels for processing.

3.2.1 **Capturing the FLIR Images** The FLIR images were stored in VHS format on a standard VHS tape. Captain Greg Tarr had previously written code on the IMAGER that allowed single frame capture of 512x512 images off of a VHS tape. Using his code, the frames were converted to images. The images were then moved onto LOUVRE through CSC (Hercules) using the following commands:

```
IMAGER to CSC from inside CSC

      COPY RAVEN::DUA1:[SROGERS.KPRIDDY]*.BIN *
            where SROGERS.KPRIDDY is the directory with the images

CSC to LOUVRE inside LOUVRE

      csc> ftp -i csc

      ftp> mget *.bin

      ftp> bye
```

The connection through CSC was necessary because there is no immediate connection to any UNIX machine from the IMAGER. Once the images were stored on LOUVRE, the 64 byte header was removed, and the images were moved to SCGRAPH using the ftp commands from above.

3.2.2 **KHOROS** **KHOROS** is a software package written and updated by the University of New Mexico. The routines available on **KHOROS** are all controlled by input and output connections of the glyphs. Using **KHOROS** the desired portion of the image was extracted and pasted onto a 512x512 black background. **KHOROS** offerred many options for target segmentation and spatial filters.

3.2.3 **Target Segmentation** The image in Figure 3 was binarized using **KHOROS**, see Figure 4. Next, a Sobel transform was performed on Figure 3 which extracted the edges extremely well. The thresholded image and the Sobel transformed image were combined using the logical AND operator. Finally, a region growing routine was used to fill in some of the choppy regions of the target, see Figure 5.

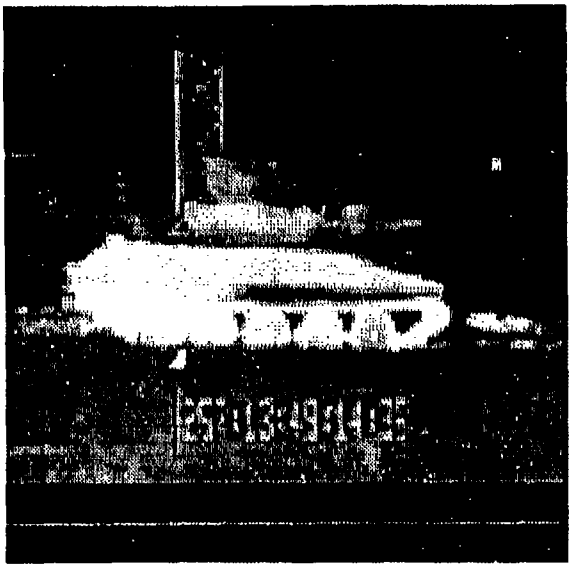


Figure 3. Original tank image. Its row and column size is 512x512 pixels.

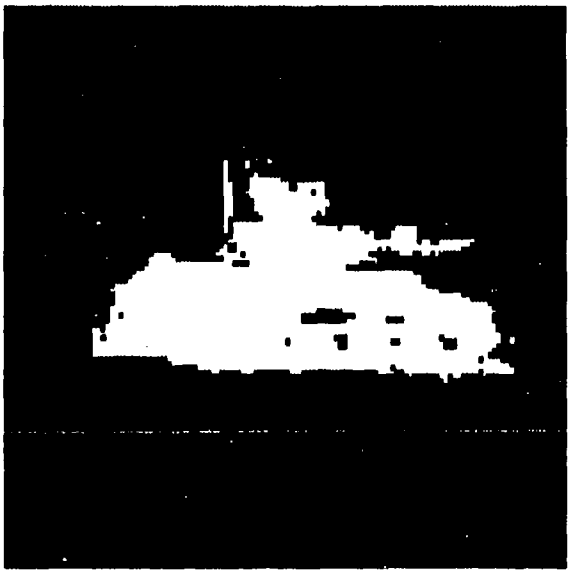


Figure 4. Binarized image of the above tank after it has been cropped and histogram equalized.

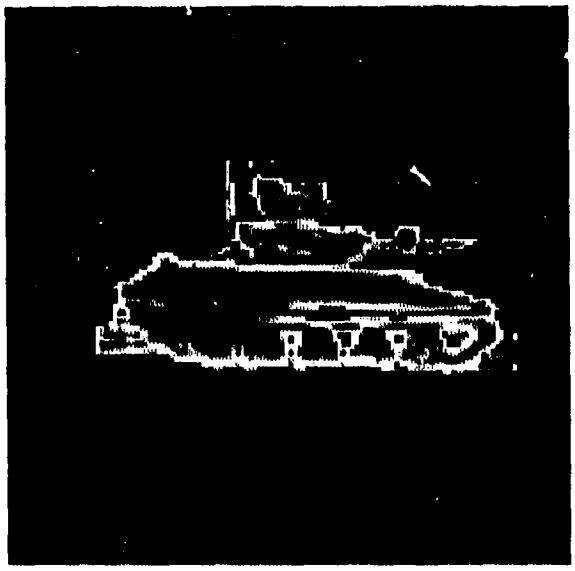


Figure 5. An image that has been combined using a Logical AND operator. The two images that were combined were a Sobel Transformed image and a histogrammed image. The final result has been enhanced using region growing.

The thresholded image, Figure 4, and the final region growing image, Figure 5, appear to be very similiar. Due to this, the shape analysis program just used the thresholded image as the truthed image of the target. To avoid the need to adjust the thresholding value each time, the cropped images were histogram equalized prior to the thresholding routine.

3.2.4 **Image Compression** Once the images were cropped and 'segmented', their size was reduced to 128x128 by using a 4 to 1 pixel averager. This was done to decrease both disk storage and image processing time. The pixel averaging scheme that the *Utah Raster Toolkit* (URT) included in its package was used for the averaging. This package can be found on the NEXT machines. Before the URT tools were used, the images were converted to RLE (Run Link Encoded) format.

3.3 Feature Set Generation

Three approaches were used to generate the feature sets: standard feature extraction, Karhunen–Loève image regeneration, and Discrete Cosine Transform regeneration. For each approach, the data sets produced were processed using **NEURAL GRAPHICS**. The results are covered in Chapter IV.

3.3.1 **Standard Feature Sets** Two directions were taken concerning standard feature set generation. The first set of features was composed of shape and intensity features while the second set consisted of Fourier components. Note: all code written for these routines was in C, and can be found in Appendix B.

3.3.1.1 **Moment Features** Moments can be used to reconstruct their images if enough are included (17 was the motivation to include moments in the feature set. The moments used in the et consisted of third order silhouette moments and the semi–major/semi–minor axes components that represent the first and second order moments. Equation 1 was used to find each moment, and Equations 2 , 3 were used to find the semi-major axis and semi-minor axis, respectively. Each of the moments were normalized by dividing each moment by target area squared, Equation 4. There was no particular justification to choosing this normalization value except its relation to the size of the target.

3.3.2 **Shape and Intensity Features** Besides the moment features, some other shape features used were

1. length to width ratio–always larger than one,
2. complexity–ratio of the # of perimeter pixels to the area of the target,
3. compactness–ratio of the area of the target to the area of a rectangle barely encompassing the entire target,
4. $perimeter^2/area$–self-explanatory,
5. boxness–ratio of target perimeter to perimeter of rectangle from above.

By incorporating the local background into the target environment, the following intensity features were calculated.

1. (max intensity - min intensity)/average intensity–self-explanatory,
2. max intensity to average intensity ratio,
3. max intensity,
4. gray level contrast–ratio of the average target intensity to the average background intensity.

The intensity features have been used before with FLIR images. In a multisensor environment, this is where FLIR imagery would contribute the most—it adds features not measurable by other sensors.

3.3.3 **Low Frequency Fourier Features** To reiterate, low frequency Fourier components can reproduce a very close approximation of an image with an inverse FFT. A 4x7 window was placed around the DC from the FFT image and the 28 components were used as features. Originally, 49 low frequency components were going to be kept, a 7x7 window about the origin. However, since the spatial images were real, only two adjacent quadrants needed to be kept from the FFT. The last two could be regenerated from the first two due to the hermitian quality of the Fourier Transform, $\mathcal{F}[]$ (6:193).

$$\begin{aligned} \mathcal{F}[\varepsilon,\eta] &= \mathcal{F}^*[-\varepsilon,-\eta] \\ \mathcal{F}[-\varepsilon,\eta] &= \mathcal{F}^*[\varepsilon,-\eta] \end{aligned}$$

To visualize this relationship, consider Figure 6. The signal from quadrant one is rotated 180° about the origin into quadrant three. And the same holds true for quadrants two and four.

3.3.4 **Karhunen–Loève Transform Using the Reduced Covariance Matrix** The reduced covariance matrix explained in Chapter II was found for two different numbers

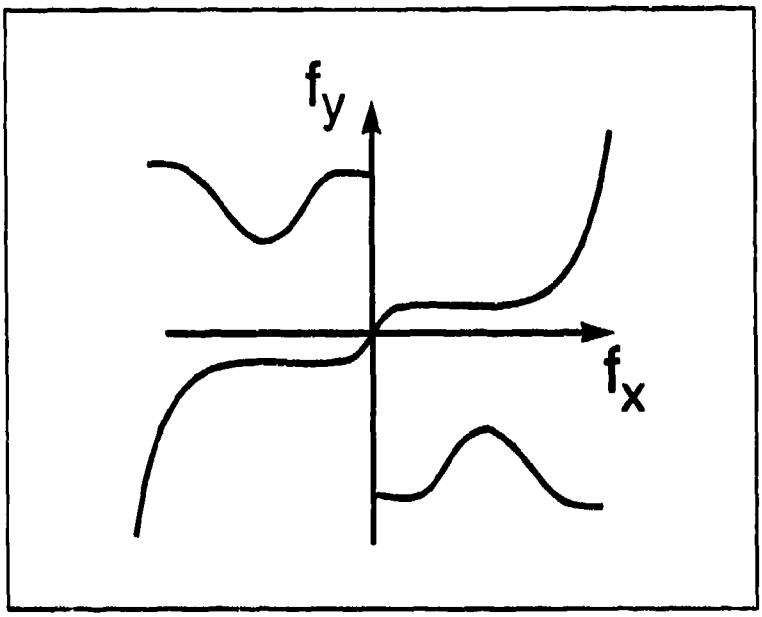


Figure 6. The odd symmetry in the Fourier domain due to a real image in the spatial domain.

of input images. These were six and nine input images, two and three from each class respectively. Eigenimages were kept or discarded according to their eigenvalues. At least 90% of the eigenvalues energy, see Equation 21, was desired for reconstruction.

$$E_{LOSS} = \sum_{j=1}^{M} \lambda_j^2 - \sum_{j=1}^{K} \lambda_j^2 \tag{21}$$

where E_{LOSS} is the energy lost in reconstruction, M is the number of total eigenvectors and K is the number kept, ($K <= M$). The reconstruction will minimize the Mean Squared Error (MSE) term:

$$MSE = \frac{1}{N^2} \sum_{i=1}^{N^2} (\mathbf{x}_i - \mathbf{x}'_i)^2 \tag{22}$$

where $\mathbf{x}$ is the original image, and $\mathbf{x}'$ is the reconstructed image. This minimalization is guaranteed by the definition of a KLT. The actual reconstruction coefficients were found by simply taking the dot product of the original images with each of the eigenimages, see

Equation 23.

$$Coeff_{i,j} = \mathbf{x}_i \mathbf{e}_j \tag{23}$$

where $Coeff_{i,j}$ is the j^{th} reconstruction coefficient for the i^{th} image, $\mathbf{x}_i$ is the i^{th} original image, and $\mathbf{e}_j$ is the j^{th} eigenimage. The reconstruction coefficients for each target were used as features in a neural net. The results are listed in Chapter IV.

3.3.5 **Discrete Cosine Transform Feature Generation** The DCT is a modified FFT where the image implemented is entirely real and symmetric so that the transform would also consist of only real values. The immediate advantage is in the computation. The phase terms turn out to be zero so that no complex math is required with this transform, see Equation 18, page 12. This makes the DCT extremely fast. The DCT produces a 128x128 transformed image of an original 128x128 image. The only quadrant from the FFT that is produced in the DCT is the first quadrant, positive frequency components only.

The DCT is based on a real, symmetric image. However, the image only has to be real, the symmetry part can be ignored. An explanation is given in Appendix A.

3.3.5.1 **Implementation** A DCT of the images will be taken, and the first 3 fundamentals in each direction will be saved as features. This corresponds to the first 3 frequency components and the DC term in each direction, including the combinations. Then, these features will be used to train and test a neural network for ATR.

3.4 Artifical Neural Network Processing

NEURAL GRAPHICS was used for all of the neural network processing (16). This program was written and updated by Captain Greg Tarr. He offerred many options with **NEURAL GRAPHICS**, including a saliency measure as well as hold-one-out testing. Both of these options were used extensively. The only neural network used for training and testing was a backprop with momentum.

3.4.1 **Backpropagation with Momentum** The backprop with momentum setup used was the same for all of the training routines. The net contained one hidden layer with seven hidden nodes, and all training iterations were set at 10,000. The network can be seen in Figure 7.

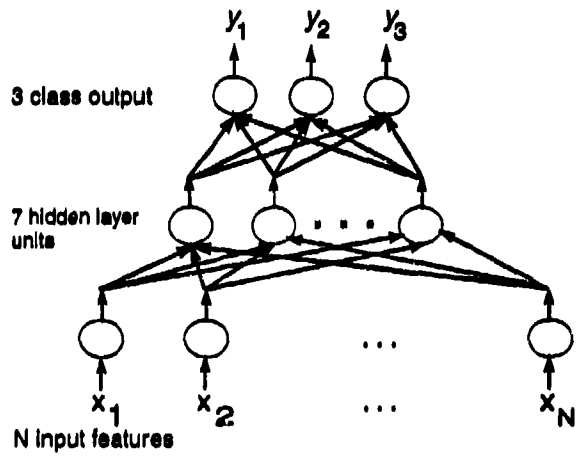


Figure 7. The 3 layer backprop with momentum network used for all neural network processing.

The backprop with momentum network was standard in the **NEURAL GRAPHICS** program. The derivation of the network itself can be found in many publications, namely Rogers' and Kabrisky's book, Ruck's dissertation, Chan's and Fallside's paper and Tarr's thesis (12, 14, 1, 16).

3.4.2 **Saliency** **NEURAL GRAPHICS** offerred a weight saliency option. This saliency measured the usefullness of an input feature with respect to the set of features that it was a member of. Once the network finished its training, an input feature's contribution to the output could be determined based on the interconnecting weights values. If a feature has very little contribution, its corresponding weights would be small numbers compared to that of an important feature's interconnecting weights. By using **NEURAL GRAPHICS'** saliency test, the top features were chosen, and this reduced feature set were tested.

3.4.2.1 **Why Saliency** Saliency is used to reduce a feature set. The advantage of reducing the feature set is twofold. First, the processing time is reduced for the neural network. Second and more important, less exemplars are needed to validate the testing results of the network. To reiterate, there are three rules that determine the accuracy of a set of test results. They are Foley's rule, Cover's rule and 'Uncle Bernie's' rule, see page 14. Using the saliency of each feature set, Foley's rule could be met without generating any more exemplars. As an example, consider the author's situation:

```
3 class output problem: tanks, jeeps and towers

# of exemplars per class:
     47 tanks,
     29 jeeps, and
     25 towers

Lowest # of exemplars is 25

Violating Cover's Rule:
     25 < 2(# of features)
     # of features > 12

Meeting Foley's Rule:
     25 >= 3(# of features)
     # of features <= 8
```

So the largest feature set possible with the above number of exemplars is 8.

3.4.3 **Hold–One–Out** Hold-one-out is a method to test data with a neural network. Since Foley says that there must be a minimum of three times the number of features as exemplars per class to train the network with, and a net should not be tested with an

exemplar it has used for training, this method gives the best results with the least amount of exemplars (14). Basically, the net is trained with all but one exemplar. When the net is trained satisfactorily, the exemplar that was held out is tested. This process is done for every exemplar, and an overall probability of correct classification is found by dividing the number of correct classifications by the total number of classifications. This method also gets around Cover's rule because the net is not trained with the exemplar it is tested with.

3.4.4 **Statistical Normalization** All the feature sets were statistically normalized prior to usage in the neural net. The statistical normalization was Gaussian, where a mean and variance were computed based on the below equations (14).

$$\begin{aligned} \bar{x}(mean) &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ Var(variance) &= \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \\ x_{norm} &= \frac{(x_i - \bar{x})}{\sqrt{Var}} \end{aligned}$$

The data was statistically normalized because one feature may consistently have values larger than one million while another may have values very close to one, and the neural network cannot account for such a dynamic range. Therefore, by keeping all the values within a range of ≈[-2,2], the network could handle each of the features equally well.

3.5 Summary

In this chapter, the approach to the research as well as an explanation of the approach were presented. The next chapter deals with the results from the research and a discussion of each.

IV. RESULTS AND DISCUSSION

4.1 Target Segmentation

Segmenting the targets was the area with the most human intervention in the entire ATR process. **KHOROS** was used to do the entire segmentation process because it offerred the best on screen interaction routines. It was impossible to automatically segment out just the target because other objects in the image also had sharp lines as well as bright spots. So the targets were hand 'cropped' and pasted elsewhere. The rest of the routine was a simple histogram equalization and threshold routine. The threshold value was set at 225 for every image. The 'thresholded' image was used as the segmented image to generate all the shape features.

4.1.1 **Cropped Images** No routine, tried by the author, could extract the target only and ignore everything else in the image. Most of the extraction routines offerred in *KHOROS* were based on thresholding and/or edge extraction, and all of the images used included other edges besides the targets. Thus, it was necessary to extract only the target from the entire image and paste it elsewhere.

4.1.2 **Histogram Equalization** A histogram equalizer was used to process the cropped image prior to thresholding it. Figure 8 shows a cropped image for each class and its corresponding 'segmented' image. The author's assumption was that by histogram equalizing the cropped image, the need to vary the thresholding value would be removed. The results after processing the features in a neural network are covered below. Due to the high recognition rates with a constant threshold value, up to 99% using the 'segmented' image, the need to vary the threshold value was removed.

4.1.3 **Problems with the FLIR Images** With IR sensors, the sensors determine the gray scale value based on a dynamic thresholding. Depending on the highest and lowest temperature values in the image, the IR sensor varies its correlation between the gray scale

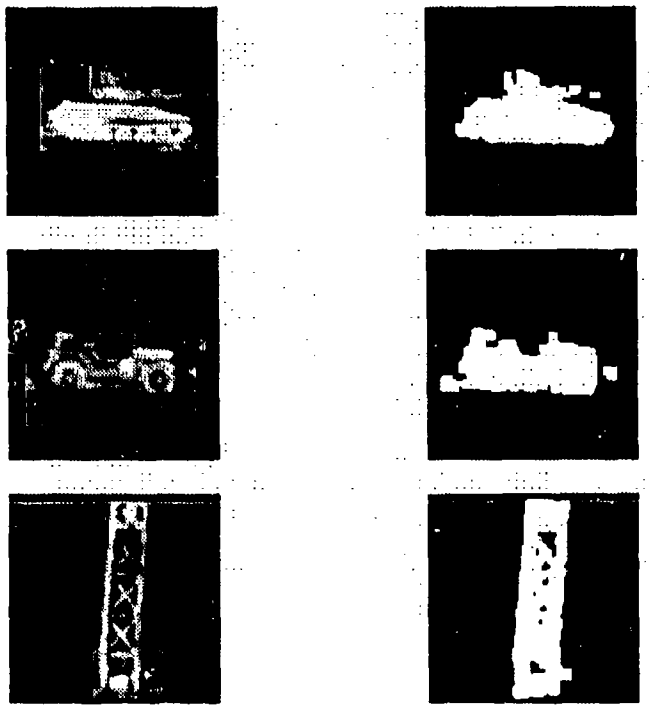


Figure 8. Cropped image (left) and its corresponding 'segmented' image (right) for each target class. The segmented images have been histogram equalized and binarized.

value and temperature step size. With older IR sensors, if a hot spot were to appear, the gray scale would be changed so much that the target would dissappear. This occurred with some of the data due to the sensor used. Captain Ken Fielding supplied some IR imagery with a new sensor and a real–time front end histogram equalizer. With this new data, the hot spots no longer made the target dissappear. Unfortunately, there was not enough time to experiment with this new data.

4.2 ATR Using Features

The feature sets mentioned in Chapter III were trained and tested using a backpropagation network with momentum. Each net contained one hidden layer consisting of seven nodes, and the testing procedure for each feature set was the hold–one–out method described in Chapter III. Six feature sets were tested that met the Foley and Cover criteria. Specifically, the smallest number of exemplars for a given class was 25. In order to meet Foley's criteria, the largest number of features in any given set was 8, see page 26. Thus, the largest feature set used consisted of 8 features, and each feature set contained 100 total exemplars: 29 jeeps, 46 tanks and 25 towers.

4.2.1 **Shape and Intensity Features** 11 shape and intensity features were used to train/test a network. The 11 features were

1. (max Intensity - min Intensity) / average Intensity,
2. max Intensity / average Intensity,
3. max Intensity,
4. semi-major axis / semi-minor axis: these can be found in Chapter II on page 6,
5. $Moment_{03}$: all the moments are silhouette moments,
6. $Moment_{30}$,
7. $Moment_{12}$,
8. $Moment_{21}$,

FEATURE	VALUE
(max Intensity - min Intensity) / average Intensity	3.23
max Intensity / average Intensity	3.57
max Intensity	0.00
semi-major axis / semi-minor axis	5.00
$Moment_{03}$	8.77
$Moment_{30}$	6.11
$Moment_{12}$	6.11
$Moment_{12}$	6.86
$Moment_{21}$	2.30
length / width	8.18
gray level contrast	1.20
$perimeter^2/area$	9.77

Table 1. Saliency of the 11 shape and intensity features. A large value means the feature is very salient.

9. length / width,
10. gray level contrast: explained on page 7,
11. $perimeter^2/area$.

Using the hold–one–out method on the aforementioned eleven features, the net recognized correctly at 100%. This result is good; however, Foley's rule was not met. So a saliency test was done on the features. The results are listed in Table 1. The top five features are listed below.

1. $perimeter^2/area$,
2. $Moment_{03}$,
3. length / width,
4. $Moment_{12}$, and
5. $Moment_{30}$.

Using the top five features, the net correctly recognized at 95.33%, see Table 2.

Run	Results
1	99.00%
2	93.00%
3	94.00%
Average	95.33%

Table 2. Results from testing the reduced feature set for shape features. The same set of features was run each time; the three runs were used to reinforce the results.

4.2.2 **Low Frequency Fourier Features** Using the 28 low frequency Fourier components as features, the net trained to 100% for target classification, and it tested at 100% using the hold–one–out method. Even though the hold–one–out method of testing was used, Foley's criteria was also not met, see page 26. So a saliency test was done, see Table 3, and the highest five Fourier components were kept. The top five Fourier features were

1. fft[0][-1],
2. fft[2][-1],
3. fft[1][-2],
4. fft[3][-1], and
5. fft[0][-3].

Using the above five features, the net tested at 89.00% correct target classification, see Table 4. So the addition of the other 23 frequencies made quite a difference in recognition rates.

Concerning the cropping of the image, the sharp boundaries around the target did produce ringing in the FFT. However, this was of no concern because only the lower frequencies were used. Ringing is predominantly a higher frequency anomaly.

FEATURE	VALUE	FEATURE	VALUE
fft[3,-3]	9.00	fft[3,-2]	11.50
fft[3,-1]	23.00	fft[3,0]	8.00
fft[3,1]	11.50	fft[3,2]	8.50
fft[3,3]	11.50	fft[2,-3]	15.00
fft[2,-2]	16.00	fft[2,-1]	25.50
fft[2,0]	7.50	fft[2,1]	18.00
fft[2,2]	9.50	fft[2,3]	20.00
fft[1,-3]	2.00	fft[1,-2]	24.00
fft[1,-1]	0.00	fft[1,0]	7.50
fft[1,1]	7.50	fft[1,2]	2.00
fft[1,3]	11.00	fft[0,-3]	20.50
fft[0,-2]	16.50	fft[0,-1]	27.00
fft[0,0]	19.50	fft[0,1]	15.00
fft[0,2]	14.50	fft[0,3]	16.50

Table 3. Saliency of the low frequency Fourier features. A large value means the feature is very salient.

Run	Results
1	89.00%
2	91.00%
3	87.00%
Average	89.00%

Table 4. Results from testing the reduced feature set for low frequency Fourier features. The same set of features was run each time; the three runs were used to reinforce the results.

Run	Results
1	99.00%
2	100.00%
3	99.00%
Average	99.33%

Table 5. Results from testing the reduced feature set for low frequency Fourier and shape features. The same set of features was run each time; the three runs were used to reinforce the results.

4.2.3 **Fusion of the Shape and Fourier Features** Using the saliency data, the top four features from each feature set were made into a new eight feature set. The basic premise here is that any target incorrectly classified by only one sensor should be picked up with the fused set. Also, any target incorrectly classified by both sets of features may have enough additional information from the fusion of the sets to correctly classify them. The features chosen are listed below.

1. $perimeter^2/area$
2. $Moment_{03}$
3. length / width
4. $Moment_{12}$
5. fft[0][-1]
6. fft[2][-1]
7. fft[1][-2]
8. fft[3][-1]

Using these eight features, the correct target classification percentage was 99.33%, see Table 5. This result meets the Foley rule for validation of test results, and it was tested with an independent set.

The fusion of the sets did produce some interesting results concerning the targets misclassified by the shape and Fourier feature sets. Using only the Fourier features, the algorithm misclassified 11 targets out of 100. The shape feature algorithm misclassified 5 targets out of 100. Only one of the targets was misclassified by both sets, and the fused feature set correctly classified this lone target. This result lends credence to the theory that the combination of features produces some extra information not available by each set alone. The fused feature set also correctly classified all the misclassified targets from the shape algorithm and all but one of the misclassified targets from the Fourier algorithm.

4.3 ATR Using Karhunen-Loève Image Reconstruction Coefficients

A reduced covariance matrix was generated using images from all three targets: tanks, jeeps and towers. Then, the Karhunen-Loève transform was taken, and the eigenvalues and eigenimages were found. The eigenvalues were representative of the usefullness of the given eigenimages in the reconstruction of the original images. So any eigenimage with a comparatively large eigenvalue would be used heavily in the reconstruction process. Comparing the eigenvalues, the eigenimages could be rank ordered by importance. The graph in Figure 9 shows the eigenvalues for a nine eigenimage routine. By keeping the first six eigenimages, $\approx$ 99% of the reconstruction energy was kept, see same plot. Based on this approach, the number of eigenimages kept was determined, see Equation 21.

Using the reduced set of eigenimages, each original image, from the entire training set, was reconstructed. For each image, a weight corresponding to each eigenimage resulted for the reconstruction. These weights were determined by taking the dot product of the original images with the new eigenimages, see Equation 23. Due to the mathematics of the KLT, the Mean Squared Error between the original image and the reconstructed image, see Equation 22, was minimized. Using these reconstruction coefficients as features in an exemplar, a neural net was than trained and tested in the same manner as the above feature sets were.

Run	Results
1	100.00%
2	100.00%
3	100.00%
Average	100.00%

Table 6. Results from testing the 6 reconstruction coefficients as features. The same set of features was run each time; the three runs were used to reinforce the results.

4.3.1 **KL Reconstruction Using Six of Nine Eigenimages** A KL transform was taken of a reduced covariance matrix made up of nine original images. The nine input images consisted of three targets per class of tanks, jeeps and towers. Of the nine eigenimages created, six were kept which accounted for ≈ 99% of the reconstruction energy, see Figure 9. Using the six reconstruction coefficients as features, the net tested at 100% correct target classification, see Table 6.

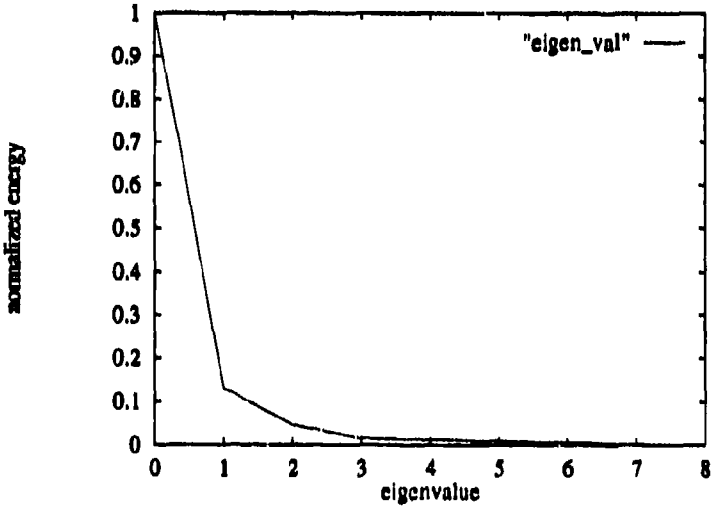


Figure 9. The eigenvalues for a nine eigenimage KLT. The energy is the magnitude of the eigenvalue squared.

The nine eigenimages produced are in Figure 10. Notice the distinct representation of

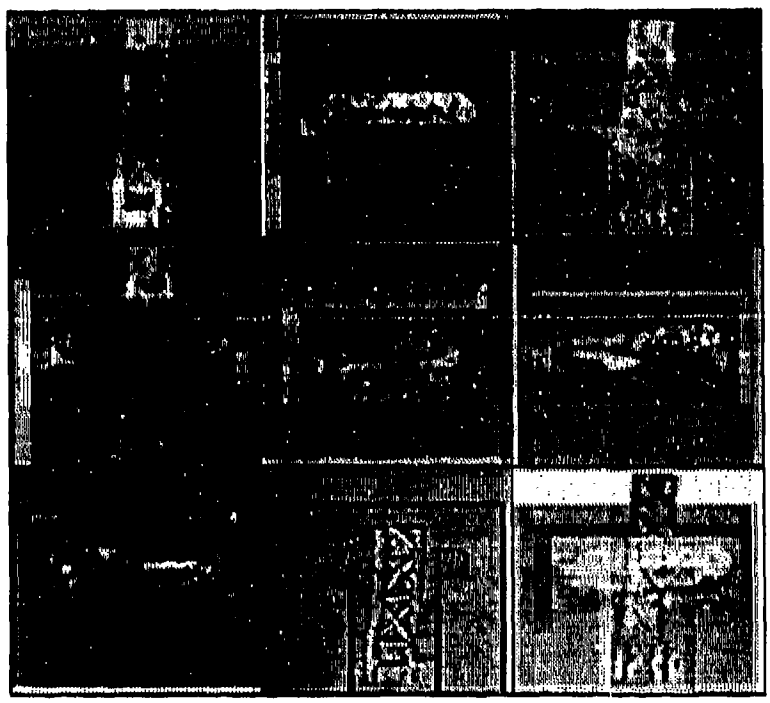


Figure 10. The 9 eigenimages of a 9 eigenimage KL Transform.

Run	Results
1	100.00%
2	100.00%
3	100.00%
4	98.00%
5	98.00%
6	98.00%
7	100.00%
8	100.00%
9	100.00%
Average	99.33%

Table 7. Results from testing the 5 reconstruction coefficients as features with three different sets of KL Transforms. The first three runs used one set of images to generate the eigenimages that was different than the next six runs. The second and third sets of three runs each used the same images to generate the eigenimages; however, the images were arranged in a different order.

the tower class in almost every eigenimage. The jeep is only apparent in one eigenimage, number 7. However, the tank class is apparent in ≈ 5 eigenimages. We would expect to see the tower and tank class reconstructions to be very accurate while the jeep class' would be difficult to see. This is explained below, see page 40. Since the results were so good, the reduced covariance matrix was created with six input images instead of nine.

4.3.2 **KL Reconstruction Using Five of Six Eigenimages** A KL transform was taken of a covariance matrix consisting of six input images, two per class of tanks, jeeps and towers. The approach was the same as the one modelled on page 35. The results of this testing can be seen in Table 7. The last six runs were based on using five out of six eigenimages from six new input images. The last KL transform was to test the reduced covariance matrix to see if it would generate the same eigenimages if the same input images were given but in a different order, and it did. The results can be seen in Figure 14. An interesting point about the middle three runs is that the network missed each time for the same two exemplars. The two tanks it incorrectly recognized are in Figure 11. As you can

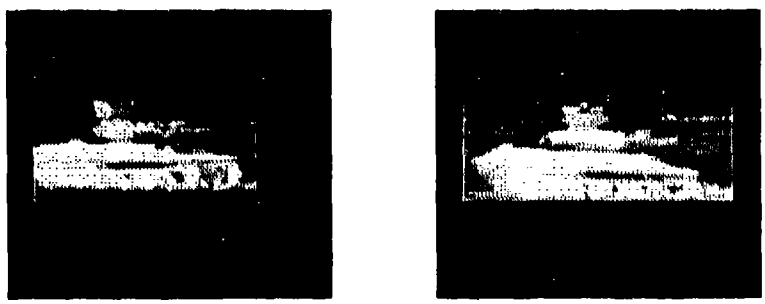


Figure 11. The two incorrectly recognized tanks for the 5/6 eigenimage KL reconstruction.

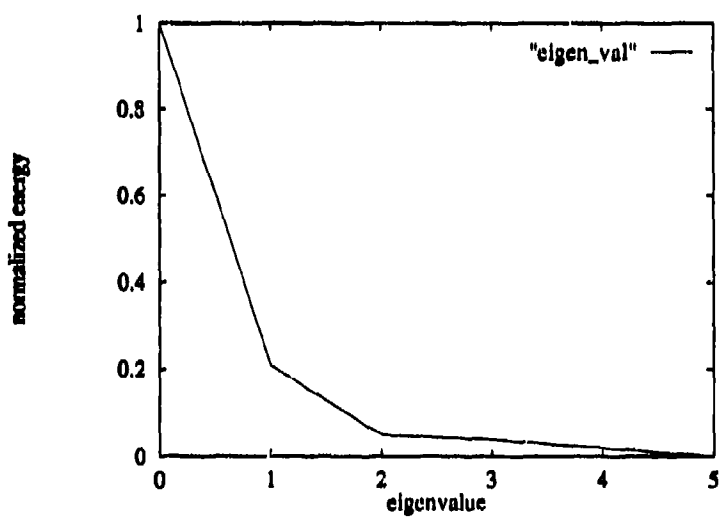
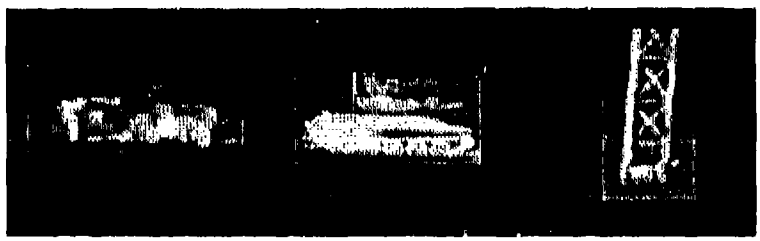


Figure 12. The eigenvalues for a six eigenimage KLT. The energy is the magnitude of the eigenvalue squared.

see, they look like tanks. The author has no deep conceptual explanation as to why the neural net couldn't correctly recognize either of these tanks.

4.3.3 **Lack of a Need for Visual Recognition to Achieve Target Recognition** By examining the reconstructed targets, the author noted a definite lack of target identification between the jeep and tank class. However, the network had no problems with determining the classification of these two targets. In Figure 13 the reconstructed targets can be seen. By examining the reconstructed images, the tower class is definitely existent



The Original Images: One per Class

The Reconstructed Images Using the Top 5 of 6 Eigenimages

Figure 13. The original and reconstructed images for each class of a 6 eigenimage KL Transform using 5 eigenimages in the reconstruction.

for the tower class and definitely nonexistent for the other two classes. This is determined by the bright tower in the reconstructed tower which means there is quite a bit of energy in that spot, and the dark tower in the other two reconstructed images which means there is

a lack of energy in that spot. Since the tank and jeep classes are not distinguishable from each other, one would surmise that targets do not have to be humanly recognizable to be computationally recognizable using the proper algorithm. Cottrell understood this when he said

> ...in a sense, neurocomputing places a higher premium on domain knowledge and network use cleverness than on deep technical understanding (10:328).

This lack of understanding adds the element of 'magic' to the neural network as well as the Karhunen-Loève Transform; however, there is none concerning each of these. The KLT simply finds the best orthogonal representation of a set of vectors, and the neural net simply determines how a set of features cluster.

4.3.4 **Incomplete Reconstruction Using the KL Eigenimages** A standard KL transform that processes the entire 16384x16384 covariance matrix, for a 128x128 image, would produce 16384 eigenimages and 16384 corresponding eigenvalues. These eigenvalues could than be rank ordered, and the corresponding reduced eigenimage set could be used for the reconstruction. The problem with this approach is that it takes a long time to process a 16384x16384 matrix! The reduced covariance matrix approach reduced the matrix to MxM, where M is the number of input images to the routine. With this compression of the covariance matrix to an approximation of it, complete reconstruction is lost; however, classification of the targets is still possible. So nothing is being sacrificed as far as classification goes which is the ultimate goal. The user does not need to see a tank as long as the missile knows what it is and where to go.

The eigenimages turned out to be orthogonal which is to be expected. This was tested by taking the dot product of any two eigenimages. If the two vectors are orthogonal, their dot product will be zero since the angle between two orthogonal vectors is always 90°, see Equation 24.

$$\cos\theta = \frac{\mathbf{ab}}{(||\mathbf{a}||)(||\mathbf{b}||)} \tag{24}$$

where $\vec{a}, \vec{b}$ are eigenvectors. However, the question remained as to whether the same eigenimages would be produced for a version of the training input images that was moved around. As it turns out, the eigenimages were the same, see Figure 14. This result is due to the determinant of the reduced covariance matrix, **L**, remaining constant.

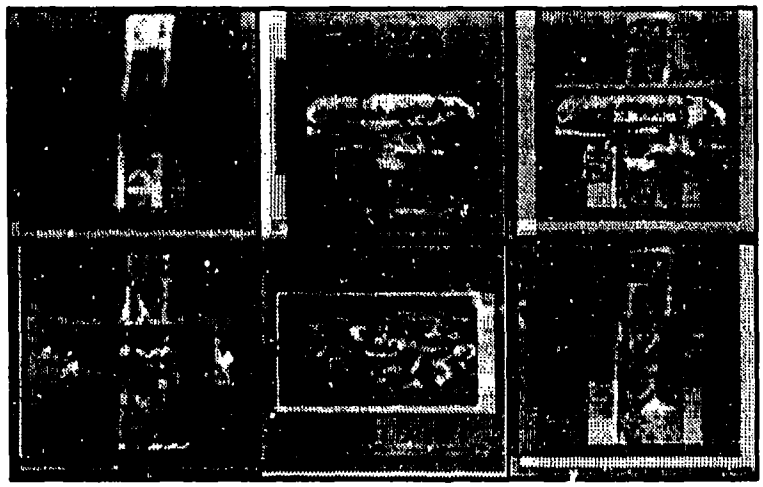
4.3.5 **KL Transform as a Glorified Correlator** Since the KL transform is very sensitive to the size of the targets as well as to their location in the image, the author believes the KL transform is only doing correlations where the correlator templates are the eigenimages. This could be a maximization of the correlation routine because the eigenimages, the templates in this case, might be the best possible templates.

4.4 ATR Using Discrete Cosine Transforms

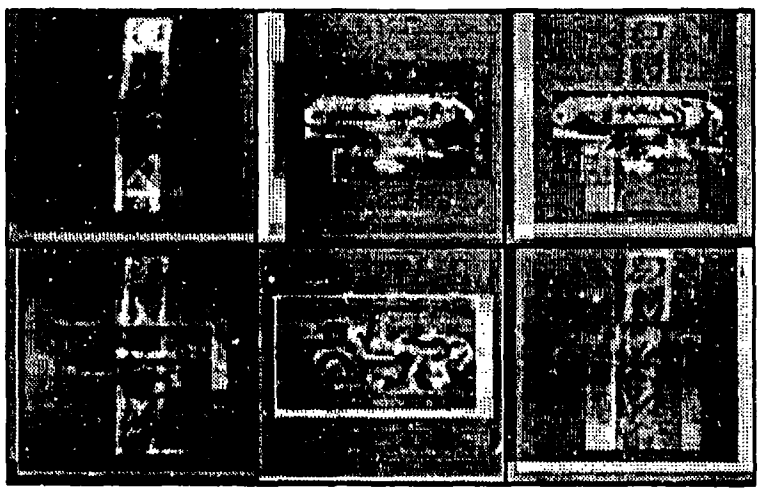
The results were positive when the DCT was used as another FFT. To reiterate, the DCT was taken of each original image, and the transform was the same size as the original image. With the DCT, a 4x4 window was taken about the origin to contain the 0^{th} through 3^{rd} harmonics. This amounted to 16 features. Using the neural net, the recognition rate was 100%, but the results are not valid. The feature space needed to be reduced to meet Foley's rule.

4.4.1 **Reduced DCT Feature Set** The saliency routine was done on the 16 features, and the results are listed in Table 8. Based on these results, the top eight features were taken to compose a new, reduced feature set listed below:

1. D[1][0],
2. D[1][1],
3. D[1][2],
4. D[1][3],
5. D[2][1],



6 Eigenimages Created from First Ordering of
6 Input Images



6 Eigenimages Created from Second Ordering of
6 Input Images Used in Both KL Transforms Above

Figure 14. The 6 eigenimages produced using the same training set only the set has been presented to the covariance matrix in a different order. Notice that the eigenimages created with the same set of six original images are the same regardless of ordering.

Feature	Value	Feature	Value
D[0][0]	0.00	D[0][1]	3.50
D[0][2]	1.50	D[0][3]	8.50
D[1][0]	15.00	D[1][1]	10.50
D[1][2]	14.00	D[1][3]	12.00
D[2][0]	3.50	D[2][1]	9.00
D[2][2]	1.50	D[2][3]	6.50
D[3][0]	12.00	D[3][1]	8.00
D[3][2]	8.00	D[3][3]	6.50

Table 8. Saliency of the 16 DCT features. A large value means the feature is very salient.

Run	Results
1	96.00%
2	94.00%
3	93.00%
Average	94.33%

Table 9. Results from testing with the top 8 DCT features. The same set of features was run each time; the three runs were used to reinforce the results.

6. D[3][0],
7. D[3][1], and
8. D[3][2].

Using these features, the net was trained and tested using the hold-one-out method. The results are listed in Table 9. The results were surprising since they should be comparable to the low frequency Fourier results. However, no direct comparisons can be made with Table 9 because the feature sets were not the same size.

4.4.2 **Comparison of the DCT Results to the Low Frequency Fourier Results** A new feature set was generated containing the top five DCT features. This was done for comparison between the low frequency Fourier results and the DCT results. The DCT

Run	Results
1	87.00%
2	89.00%
3	87.00%
Average	87.67%

Table 10. Results from testing with the top 5 DCT features. The same set of features was run each time; the three runs were used to reinforce the results.

results are listed in Table 10, and the FFT results are listed in Table 4. The FFT produced better results than the DCT, 89.00% versus 87.67% for the DCT. However, the difference between the two is pretty small. Again, each percentage was averaged from three separate runs to determine the correct classification using hold–one–out testing.

As it turns out, the top five FFT features were

1. fft[0][-1],
2. fft[2][-1],
3. fft[1][-2],
4. fft[3][-1], and
5. fft[0][-3].

And, the top five DCT features were

1. D[1][0],
2. D[1][2],
3. D[1][3],
4. D[3][0], and
5. D[1][1].

Approach	Results
TOP 5 SHAPE FEATURES	95.33%
TOP 5 FFT FEATURES FROM 28 LOW FREQUENCIES	89.00%
8 FEATURES: TOP 4 SHAPE FEATURES AND TOP 4 FFT FEATURES	99.33%
6 RECONSTRUCTION COEFFICIENTS FROM 6 OF 9 EIGENIMAGES	100.00%
5 RECONSTRUCTION COEFFICIENTS FROM 5 OF 6 EIGENIMAGES	99.00%
TOP 8 DCT FEATURES	94.33%
TOP 5 DCT FEATURES	87.67%

Table 11. Each approach to ATR and its correct classification percentages.

It was not surprising that the saliency routine for the DCT picked almost the exact same features as those from the saliency routine for the FFT. Each reduced feature set includes one third harmonic in each direction where the counterharmonic is a first harmonic or a DC. There are also terms including a first harmonic in one direction with the other direction being a DC term. The similarity between the sets is quite relieving. The author would have been worried if there appeared to be no correlation between the two reduced feature sets. Also, the results from using those particular components are so close, 89.00% for the FFT and 87.67% for the DCT, that one would surmise each is equivalent.

4.5 Comparison of All the Approaches to Target Recognition

Many approaches were tried to classify tanks, jeeps and towers. The main three areas were feature set generation using shape and FFT features, KL Transform reconstruction coefficients and DCT coefficients. Here the results will be recapped concerning the percentages of correct target classification for each approach and its variations.

It appears that the best approaches were the KL Transform reconstruction coefficients using six of nine, 100.00%, and the top eight shape/FFT features determined by a saliency routine, 99.33%. The shape/FFT approach would be faster because the only preprocessing would be a histogram equalization/thresholding routine. However, if the eigenimages were already stored, a dot product of the original image with each eigenimage would be fast for

the reconstruction routine as well.

V. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This thesis has presented some very interesting results. The work done with the eigenimages opened up a new area of ATR. The results prove that the reduced covariance matrix that Suarez and Turk and Pentland used for face recognition can be just as powerful for other targets as well (15, 18). More importantly, the need for three separate KL Transforms, one for each class of targets, was not found. The recognition rates of 100% for the combined class transform demonstrates this. Also, these results can be compared directly to standard feature extraction implementation. Using saliency routines to pick the best, the recognition rates were as high as 99.33%. These are also outstanding results in and of themselves considering the assumption of having segmented targets was not made. The only user interaction was in the cropping of the targets. The rest of the segmenting to classification stages was automatic. Now one point needs to be reiterated: the KL Transform mathematics is just doing a one time correlation between the original image and the eigenimages. The combination of how much each eigenimage looks like the input image determines the reconstruction coefficients and, apparently, separates the target classes in some space where the neural net could classify them. Note: in a real-time application, the covariance matrix would not have to be generated, the eigenimages would already be stored, and these would be used for the reconstruction.

The DCT was also evaluated and compared to an identical FFT application. The DCT results were very close to those of the FFT when the low frequency components from each set were kept. Also, a saliency routine done on both the low frequency Fourier feature sets proved that certain components are definitely more beneficial than others regardless of how the Fourier features were generated. The best recognition by the DCT with 8 features was 94.33%.

The backprop with momentum network used in all of the feature processing worked very well. It was a fast network due to the single hidden layer with only seven nodes, and the training iterations being set to 10,000 before the net was tested. Also, the hold-one-out testing method proved to work very well. It offerred the best approach to meeting Foley's criteria of validating the test results.

5.2 Recommendations

5.2.1 **Cottrell** Since the KL transforms worked so well, a Cottrell network would be an obvious next step. Cottrell's 'holons' would replace the 'eigenimages' from the KLT and could be implemented much as the KLT was.

5.2.2 **Karhunen–Loève Transform** The KLT proved very successful in target classification; however, it is still dependent upon scale, shift and rotation. Maybe a routine to make the input images scale, rotation and shift invariant could be implemented before the KLT stage. Assuming the PSRI stage would not take too long, this could make a complete target classifier with real-time applications.

Also, as a preliminary step to make the classifier user independent, concerning cropping the images, the entire image could be scanned a section at a time using the previous KL transform except with an additional class: non-target. This could prove very useful for the above idea in that it would remove the need to make the images shift invariant.

5.2.3 **Discrete Cosine Transform** The DCT in the author's research was implemented much the same as any FFT has been implemented: to generate low frequency Fourier features. However, Robert Gray's efforts with the DCT show that eigenvectors and eigenvalues can be generated from a covariance matrix that is some permutation of the DCT values (9). This could be pursued and implemented as the KLT was.

5.2.4 **Multisensor Applications** The wave of the future in ATR is directed towards multisensor applications. Many feel that it is the best method of modelling the

human visual system, and the author agrees. Simply, one sensor can offer advantages that another sensor cannot. In other words, one sensor's downfall may be another's trump. So the multisensor direction should definitely be pursued, and the KLT could be used with the FLIR imagery while another transform might work better with another sensor.

Appendix A. FFT Implementation of the DCT

This Appendix presents a method of taking the DCT using only an N-point FFT.

Makhoul demonstrated a method of taking the DCT of an N-point sequence using a $2N$-point FFT of a reordered version of the original sequence (11). To take the DCT of an N-point real data sequence $x(n)$, $0 \leq n \leq N-1$ first define a $2N$-point even extension of $x(n)$

$$y(n) = \begin{cases} x(n), & 0 \leq n \leq N-1 \\ x(2N-n-1), & N \leq n \leq 2N-1 \end{cases} \tag{25}$$

Figure 15 shows $x(n)$ and it's even extension $y(n)$ as defined in equation 25. Note that $y(2N-n-1) = y(n)$.

The DFT of $y(n)$ is given by

$$Y(k) = \sum_{n=0}^{2N-1} y(n) W_{2N}^{nk} \tag{26}$$

Where $W_M = e^{-i2\pi/M}$ (11:28). Substituting equation 25 into equation 26 yields

$$Y(k) = \sum_{n=0}^{N-1} x(n) W_{2N}^{nk} + \sum_{n=N}^{2N-1} x(2N-n-1) W_{2N}^{nk} \tag{27}$$

Changing the summation variable in the right-hand term, noting that $W_{2N}^{2mN} = 1$ for m an integer, and factoring out $W_{2N}^{-k/2}$ yields

$$Y(k) = W_{2N}^{-k/2} 2 \sum_{n=0}^{N-1} x(n) [W_{2N}^{nk} W_{2N}^{k/2} + W_{2N}^{-nk} W_{2N}^{-k/2}] \tag{28}$$

Equation 28 can be written as

$$Y(k) = W_{2N}^{-k/2} 2 \sum_{n=0}^{N-1} x(n) \cos \frac{\pi(2n+1)k}{2N}, \quad 0 \leq k \leq 2N-1 \tag{29}$$

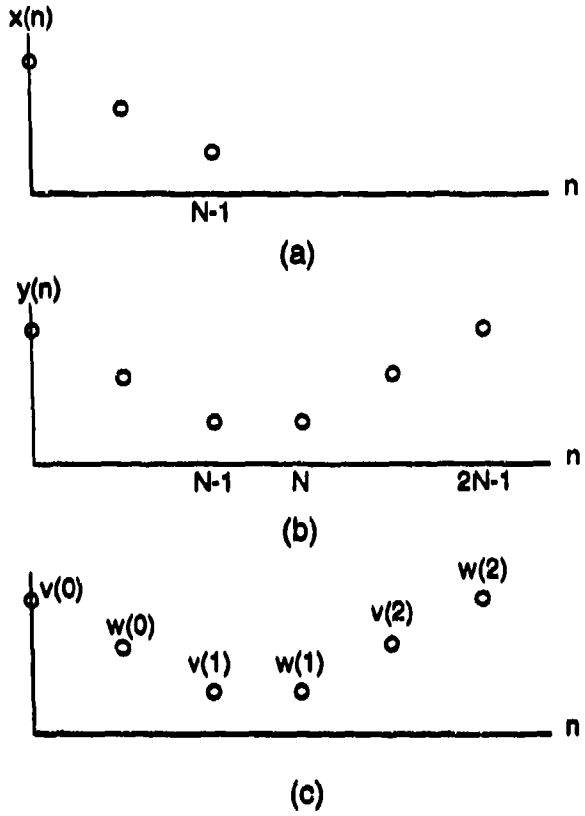


Figure 15. (a) Causal signal x(n). (b)Even extension of x(n), y(n). (c) Division of y(n) into its even and odd parts v(n) and w(n).

or

$$Y(k) = W_{2N}^{-k/2} 2Re\left[W_{2N}^{k/2} \sum_{n=0}^{N-1} x(n) W_{2N}^{nk}\right], \quad 0 \leq k \leq 2N-1 \tag{30}$$

Making a slight change in the magnitude term of the DCT definition in equation 31 yields

$$G_x(k) = \frac{2}{M} \sum_{m=0}^{M-1} X(m) \cos \frac{(2m+1)k\pi}{2M}, \quad \text{k = 1,2,...,M-1} \tag{31}$$

$$C(k) = 2 \sum_{n=0}^{N-1} x(n) \cos \frac{\pi(2n+1)k}{2N}, \quad 0 \leq k \leq N-1 \tag{32}$$

using equation 29 and equation 32 gives

$$Y(k) = W_{2N}^{-k/2} C(k) \tag{33}$$

$$C(k) = W_{2N}^{k/2} Y(k) \tag{34}$$

and, from equation 30 and equation 33

$$C(k) = 2Re\left[W_{2N}^{k/2} \sum_{n=0}^{N-1} x(n) W_{2N}^{nk}\right] \tag{35}$$

Thus, the DCT of $x(n)$ by taking the $2N$-point FFT of $y(n)$, which equation 35 demonstrates is equivalent to equation 37 except for magnitude (11:29).

$$G_x(0) = \frac{\sqrt{2}}{M} \sum_{m=0}^{M-1} X(m) \tag{36}$$

$$G_x(k) = \frac{2}{M} Re\left\{e^{-ik\pi/2M} \sum_{m=0}^{2M-1} X(m) W^{km}\right\} \tag{37}$$

Where $W = e^{-i2\pi/2M}$, $i = \sqrt{-1}$, and $X(m) = 0$, for $m = M, (M+1), \ldots, (2M-1)$. Now to demonstrate that this can be further reduced to an N-point FFT.

First divide $y(n)$ into two N-point sequences

$$v(n) = y(2n), \quad 0 \leq n \leq N-1 \tag{38}$$

$$w(n) = y(2n+1), \quad 0 \leq n \leq N-1 \tag{39}$$

where $v(n)$ and $w(n)$ are the even and odd points of the sequence $y(n)$ as labeled in figure 15(c). Note that both $v(n)$ and $w(n)$ contain all of the samples of the original sequence $x(n)$. It can be shown using equations 25, 38 and 39 that

$$w(n) = v(N-n-1), \quad 0 \leq n \leq N-1 \tag{40}$$

Now, substituting equation 38 and 39 into equation 26 yields

$$Y(k) = \sum_{n=0}^{N-1} v(n) W_{2N}^{2nk} + \sum_{n=0}^{N-1} w(n) W_{2N}^{(2n+1)k} \tag{41}$$

Now, substituting equation 40 into equation 41, noting that $W_{2N}^{2nk} = W_N^{nk}$, massaging terms, and using equations 33 and 34 yields

$$C(k) = 2Re\left[W_{4N}^{k} \sum_{n=0}^{N-1} v(n) W_N^{nk}\right], \quad 0 \leq k \leq N-1 \tag{42}$$

which may be written as

$$C(k) = 2 \sum_{n=0}^{N-1} v(n) \cos \frac{(4n+1)k\pi}{2N}, \quad 0 \leq k \leq N-1 \tag{43}$$

which is a valid alternate definition for the DCT (11:30).

Appendix B. C Code

All code used for this research was written in C. Some of the code was written by other students, and can be found in their theses. Specifically, the Karhunen–Loeève Transform code can be found in Captain Pedro Suarez' thesis(15), and the DCT code can be found in Captain Jim Goble's thesis(7). The KLT code was changed only in that the average image value for each pixel was not added back in to the reconstructed image. The DCT code was adjusted to take the DCT of a 128x128 image.

MM19.C–takes the FFT, shape and intensity features. Needs MOMENT.C, FFT1.C, FOURN.C, DOFLIP.C and NRUTIL.C.

```
/*=======================================*/
/*=======================================*/
/*=====  FEATURE GENERATOR  ==========*/
/*=====  2LT BRIAN SINGSTOCK  ==========*/
/*=====  ADVISOR: MAJOR ROGERS ==========*/
/*=======================================*/
/*=======================================*/

/*=======================================*/
/*======== SYSTEM INCLUDES ==============*/
/*=======================================*/

#include <stdio.h>
#include <math.h>
#include <string.h>

/*=======================================*/
/*======== DEFINES ======================*/
/*=======================================*/

#define ROW       128
#define COLUMN        128
#define SQ(A)       (A*A)
#define loopi(X)       for (i=1; i≤X; i++)
#define loopj(X)    for (j=1; j≤X; j++)
#define loopk(X)    for (k=1; k≤X; k++)
#define max(a,b)    (a > b) ? a : b;
#define min(a,b)    (a < b) ? a : b;

/*=======================================*/
/*=====  MAIN ROUTINE  ==================*/
/*=======================================*/

void main(argc, argv)
int argc;
char *argv[];
{

/*=======================================*/
/*===  PROGRAM INCLUDES  ================*/
/*=======================================*/

void doflip();
void fourn();
void fft1();
void nrutil();
void moment();
```

```
/*=========================================*/
/*=====  LOCAL VARIABLES  ================*/
/*=========================================*/

int i, j, k, c;
int **tru, **dat, **imatrix();
int tgt_type, TRUTH, maxI, minI;
int area, perimeter, nn[2], count;
float *mag, *vector(), glc, avgI, tI, bI;
float ab, mom30, mom03, mom21, mom12, xdivy;
char trufile[50], datfile[50];
FILE *ftru, *fdat, *fout, *fopen();

/*=========================================*/
/*=====  EXPLAIN USAGE, IF PROBLEM  =======*/
/*=========================================*/

   if (argc ≠ 2)
   {
      printf("\n\nUsage: mm19 filename wo/extension\n\n");
      exit(10);
   }

/*=========================================*/
/*== CALL HEADER AND BEGIN MANIPULATION  ==*/
/*=========================================*/

   printf("Inside of mm19.c for %s\n", argv[1]);

/*=========================================*/
/*= READ IN TRUTHED AND DATA VALUES INTO VECTORS =*/
/*=========================================*/

   sprintf(trufile,"%s-tru.dat", argv[1]);

   sprintf(datfile, "%s.dat", argv[1]);

/*=============================================== =*/
/*= CHECK TO SEE WHAT THE TARGET TYPE IS FROM THE NAME =*/
/*= OF THE INPUT STRING. IF IT ISN'T ONE OF THE ONES   =*/
/*= BELOW THAN EXIT CAUSE IT MAY BE A TRUTH FILE !!!!  =*/
/*=============================================*/

   if ((strlen(argv[1])) ≥ 9)
   {
      printf("Exiting program for %s: truth file\n", argv[1]);
      exit(100);
   }
   else if (datfile[2]=='a')
   {
```

```
    tgt_type = 1;
}
else if (datfile[2]=='e')
{
    tgt_type = 2;
}
else if (datfile[2]=='o')
{
    tgt_type = 3;
}
else
{
    tgt_type = 0;
}

TRUTH = 1;

if(!(ftru = fopen(trufile, "rb")))
{
    TRUTH = 0;
}


if (!(fdat = fopen(datfile, "rb")))
{
    fclose(ftru);
    exit(100);
}

if (!(fout = fopen("feature_fft.dat", "ab")))
{
    fclose(ftru);
    fclose(fdat);
    exit(100);
}

dat = imatrix(1, ROW, 1, COLUMN);
tru = imatrix(1, ROW, 1, COLUMN);
mag = vector(1, 28);

loopi(ROW)
{
    loopj(COLUMN)
    {
        fscanf(ftru, "%d  ", &tru[i][j]);
        fscanf(fdat, "%d  ", &dat[i][j]);
    }
}

fclose(ftru);
```

```
   fclose(fdat);

   loopi(ROW)
   {
      loopj(COLUMN)
      {
         if (tru[i][j] ≤ 10)
         {
            tru[i][j] = 0;
         }
         else
         {
            tru[i][j] = 1;
         }
      }
   }

/*=============================================*/
/*==  NOW FIND SOME FEATURES  ==============*/
/*=============================================*/

   /*====FIND INTENSITY STUFF ============*/

   maxI = 0;
   minI = 0;
   avgI = 0.0;

   loopi(ROW)
   {
      loopj(COLUMN)
      {
         c = dat[i][j];
         avgI += (float)(c);

         maxI = max(c, maxI);

         minI = min(c, minI);

      }
   }

   avgI /= (float)(ROW + COLUMN);

/*    fprintf(fout,"1 %f %f %d ",(float)(maxI-minI)/avgI,
         (float)(maxI)/avgI, maxI);  */

/*=========================================*/
/*==  THE NEXT SECTION IS DEDICATED TO FINDING THE ==*/
/*== 7x7 FOURIER COMPONENTS, KEEP 28 CAUSE REAL IMAGE ==*/
/*=========================================*/
```

```
   /*== FIND THE FFT OF THE IMAGE ==*/

   fft1(mag, dat, ROW);

   /*===== PRINT TO FILE =====*/

   fprintf(fout, "1 ");

   loopi(28)
   {
      fprintf(fout, "%f ", mag[i]);
   }

   /*=====   SKIP MOMENT STUFF CAUSE FOURIER FILE   ======*/

   TRUTH = 0;

/*=========================================*/
/*= IF TRUTHED DATA WAS AVAILABLE, FIND SOME OTHER STUFF =*/
/*=========================================*/

   if (TRUTH)
   {

      /*== FIND THE MOMENT STUFF ======*/
      /*== A, B and THIRD ORDER  ======*/

      moment(tru, ROW, &ab, &mom03,
            &mom30, &mom12, &mom21, &xdivy);

      ab = max(ab, 1.0/ab);

/*       xdivy = max(xdivy, 1.0/xdivy);  */

      fprintf(fout,"%f %f %f %f %f %f ", ab, mom03, mom30,
            mom12, mom21, xdivy);

      area = 0;
      perimeter = 0;
      count = 0;
      tI = 0.0;
      bI = 0.0;

      for (i=2; i<COLUMN; i++)
      {
         for (j=2; j<ROW; j++)
         {

            tI += (float)(tru[i][j]);
            bI += (float)(dat[i][j]);
```

```
            if (tru[i][j])
            {
               area++;
               count++;

               if ((!(tru[i-1][j]))||(!(tru[i][j-1]))
              ||(!(tru[i+1][j]))||(!(tru[i][j+1])))
               perimeter++;

            }
         }
      }
      bI -= tI;
      bI /= (float)(ROW + COLUMN - count);
      tI /= (float)(count);
      glc = tI / bI;  /*== GREY LEVEL CONTRAST ==*/
      fprintf(fout, "%f %f ", glc, (float)(SQ(perimeter))          / (float)(area));
   }
   fprintf(fout, "%d\n\n", tgt_type);
   fclose(fout);

/*=============================================*/
/*== CLOSE UP THE MATRICES OPENED  ========*/
/*=============================================*/

/*   free_imatrix(dat, 1, ROW, 1, COLUMN);
   free_imatrix(tru, 1, ROW, 1, COLUMN);
   free_vector(mag, 1, 28);  */

   printf("FINISHED\n");

/*=============================================*/
/*== GOOD NIGHT, GOOD BYE, AND GOOD RIDDANCE ==*/
/*=============================================*/

}
```

MOMENT.C–takes the silhouette moments of the input object.

```
/*=================================================*/
/*======== SYSTEM INCLUDES ========================*/
/*=================================================*/

#include <stdio.h>
#include <math.h>
#include <string.h>
/*=================================================*/
/*============ DEFINES ============================*/
/*=================================================*/

#define SQ(A)        (A*A)
#define loopi(X)       for (i=1; i≤X; i++)
#define loopj(X)    for (j=1; j≤X; j++)
#define loopk(X)    for (k=1; j≤X; j++)
#define max(a,b)    (a > b) ? a : b;
#define min(a,b)    (a < b) ? a : b;

/*=================================================*/
/*=== ROUTINE TO FIND A, B, AND THIRD ORDER MOMENTS ===*/
/*=================================================*/

void moment(tdata, ROW, ab, mom03, mom30, mom12, mom21,
      x_y)

int **tdata, ROW;
float *x_y, *ab, *mom30, *mom03, *mom21, *mom12;

{

   int ix, iy, x, y, COLUMN, area;
   int max_width, max_height, max_x, min_x, max_y, min_y;
   int CX, CY, i, j, k;
   double bsm20, bsm02, bsm11, bsm30, bsm03, bsm12, bsm21;
   double normal2, normal3, average_y;
   double tx, ty, a, b, x_shift, y_shift, average_x;

   COLUMN = ROW;
   CX = COLUMN/2;
   CY = ROW/2;


      /*=== find center of image ===*/

   ix = iy = 0;

   for (y = 1; y ≤ ROW; y++)
   {
      for (x = 1; x ≤ COLUMN; x++)
      {
         if (tdata[y][x])
```

```
        {
            ix += x;
            iy += y;
        }
    }
}

average_x = (double)(ix / area);
average_y = (double)(iy / area);

/*=== find amount to shift data for shift-invariance ===*/

x_shift = average_x - (double)(CX);
y_shift = average_y - (double)(CY);

    /*=== find Binary-Shape-Moments (BSM) ===*/

bsm20 = bsm11 = bsm02 = bsm30 = bsm03 = bsm21 = bsm12 = 0.0;

for (y = 1; y ≤ ROW; y++)
{
    for (x = 1; x ≤ COLUMN; x++)
    {
        if (tdata[y][x])
        {
            tx = (double)(x) - x_shift;
            ty = (double)(y) - y_shift;

            bsm20 += tx * tx;
            bsm11 += tx * ty;
            bsm02 += ty * ty;

            bsm30 += bsm20 * tx;
            bsm03 += bsm02 * ty;

            bsm21 += bsm20 * ty;
            bsm12 += bsm02 * tx;
        }
    }
}


    /*=== calculate normalization constants ===*/

normal2 = (double)area;
normal3 = (double)(area) * (double)(area);


/*=========================================================*/
/*=========================================================*/
 /*= Find the a and b values of the ellipse    ====*/
```

```
/*====  from the third order moments          ====*/
/*=============================================================*/
/*=============================================================*/

a = sqrt((bsm20+bsm02+sqrt(SQ(bsm20-bsm02)+
   4*bsm11*bsm11))/area/2.0);
b = sqrt((bsm20+bsm02-sqrt(SQ(bsm20-bsm02)+
   4*bsm11*bsm11))/area/2.0);

/*=============================================================*/
/*===  calculate the maximum horizontal        ===*/
/*===  and vertical chords                     ===*/
/*=============================================================*/

min_x = COLUMN;     /* set range one past opposite ends */
max_x = -1;

max_width = 0;

for (y = 1; y ≤ ROW; y++)
{
   for (x = 1; x ≤ COLUMN; x++)
   {
      if (tdata[y][x])
      {
         min_x = min(x, min_x);
         max_x = max(x, max_x);
      }

   }
   max_width = max(max_width, max_x - min_x +1);
}

   /*===  calculate maximum vertical chord  ===*/

min_y = ROW;  /* set range one past opposite ends */
max_y = -1;

max_height = 0;

for (x = 1; x ≤ COLUMN; x++)
{
   for (y = 1; y ≤ ROW; y++)
   {
      if (tdata[y][x])
      {
         min_y = min(y, min_y);
         max_y = max(y, max_y);
      }
   }
```

```
        max_height = max(max_height, max_y - min_y +1);
    }

    *ab = (float)(a/b);
    *mom30 = (float)(bsm30/normal3);
    *mom03 = (float)(bsm03/normal3);
    *mom21 = (float)(bsm21/normal3);
    *mom12 = (float)(bsm12/normal3);
    *x_y = (float)(max_width)/(float)(max_height);
    printf("%f %f %f %f %f\n",*ab,*mom30,
          *mom03,*mom21,*mom12);

}
/*=============================================================*/
/*==========   THE END !!!!       ====================*/
/*=============================================================*/
```

FFT1.C–takes the FFT of the input object. Needs DOFLIP.C and FOURN.C.

```
/*=========================================*/
/*======  SYSTEM INCLUDES ================*/
/*=========================================*/

#include <stdio.h>
#include <math.h>
#include <string.h>

/*=================================================*/
/*== FFT ROUTINE THAT IS SIZE FLEXIBLE          ==*/
/*==  JUST INPUT THE LENGTH AND WIDTH OF THE IMAGE ==*/
/*==  TO BE FFT'D. THE OUTPUT IS THE UPPER HALF OF ==*/
/*==  A 7 BY 7 LOW FREQ FOURIER BOX-REAL IMAGE    ==*/
/*==                                          ==*/
/*==      2LT BRIAN SINGSTOCK                 ==*/
/*==        9 AUG 1991                        ==*/
/*==         MAJ ROGERS                       ==*/
/*==                                          ==*/
/*=================================================*/

/*=========================================*/
/*==========  DEFINES  ====================*/
/*=========================================*/

#define SQ(A)       (A*A)
#define loopi(X)      for (i=1; i≤X; i++)
#define loopj(X)    for (j=1; j≤X; j++)
#define loopk(X)    for (k=1; j≤X; j++)
#define max(a,b)    (a > b) ? a : b;
#define min(a,b)    (a < b) ? a : b;

void fft1(mag, dat, ROW)
int **dat, ROW;
float *mag;

{

   void doflip();
   void fourn();

   int count, i, j, nn[2], LENGTH;
   float *fft, *vector(), dummy;

   LENGTH = ROW;

   count = 1;

   fft = vector(1, LENGTH*LENGTH*2);

   loopi(LENGTH*LENGTH*2)
```

```
    {
       fft[i] = 0.0;
    }

    loopi(LENGTH)
    {
       loopj(LENGTH)
       {
          fft[(LENGTH*2*(i-1))+(j*2-1)] = dat[i][j];
       }
    }

    /*= NOW FFT IS ZERO PADDED AND COMPLEX FORMAT IS ACCT'D FOR =*/

    nn[0] = LENGTH;
    nn[1] = nn[0];

    fourn(fft, nn, 2, 1);

    doflip(fft, LENGTH);

/*=============================================================*/
/*== now take the magnitude of the complex response ==*/
/*== and send the 28 components for a 4x7 box to   ==*/
/*== the feature file                              ==*/
/*=============================================================*/

    count = 1;

    for (i=LENGTH/2-3; i<=LENGTH/2; i++)
    {
       for (j=LENGTH-7; j<=LENGTH+5; j+=2)
       {
          mag[count] = sqrt(SQ(fft[i*2*LENGTH+j])
                + SQ(fft[i*2*LENGTH+j+1]));
          count++;
       }
    }

    free_vector(fft, 1, LENGTH*LENGTH*2);
}
```

Bibliography

1. Chan, L.W. and F. Fallside. "An adaptive training algorithm for back propagation networks," *Computer Speech and Language*, pages 205–18 (1987).
2. Cottrell, Garrison W. and Janet Metcalfe. "EMPATH: Face, Emotion, and Gender Recognition Using Holons." Institute for Neural Computation, University of California San Diego.
3. Cover, Thomas M. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition," *IEEE Transactions on Electronic Computers*, *EC*-(14):326–334 (June 1965).
4. Fleming, Michael K. and Garrison W. Cottrell. "Categorization of Faces Using Unsupervised Feature Extraction," *IEEE International Joint Conference on Neural Networks*, pages 65–70 (1990).
5. Foley, Donald H. "Considerations of Sample and Feature Size," *IEEE Transactions on Information Theory*, *IT-18*:618–626 (September 1972).
6. Gaskill, Jack D. *Linear Systems, Fourier Transforms and Optics*. New York: John Wiley and Sons, 1978.
7. Goble, James R. *Face Recognition using the Discrete Cosine Transform*. MS thesis, Air Force Institute of Technology, Wright–Patterson Air Force Base, December 1991.
8. Gonzalez, Rafael C. and Paul Wintz. *Digital Image Processing*. Reading, Ma: Addison–Wesley, 1987.
9. Gray, Robert Molten. "On the Asymptotic Eigenvalue Distribution of Toeplitz Matrices," *IEEE Transactions on Information Theory*, pages 725–730 (November 1972).
10. Hecht-Nielson, Robert. *Neurocomputing*. Reading, Mass: Addison-Wesley Publishing Company, 1990.
11. Makhoul, John. "A Fast Cosine Transform in One and Two Dimensions," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 27–34 (February 1980).
12. Rogers, Steven K. and Matthew Kabrisky. *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition*. Bellingham, Wa: SPIE, 1990.
13. Roth, Michael W. "Survey of Neural Network Technology for Automatic Target Recognition," *IEEE Trans. for Neural Networks*, *1*(1):28–43 (March 1990).
14. Ruck, Dennis W. *Characterization of Multilayer Perceptrons and their Application to Multisensor Automatic Target Recognition*. PhD dissertation, Air Force Institute of Technology, 1990.
15. Suarez, Pedro F. *Face Recognition with the KL Transform*. MS thesis, Air Force Institute of Technology, Wright–Patterson Air Force Base, December 1991.

16. Tarr, Gregory L. *Dynamic Analysis of feedforward Neural Networks Using Simulated and Measured Data*. MS thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, December 1988.

17. Teague, Michael Reed. "Image Analysis Via the General Theory of Moments," *Optical Society of America*, pages 920-30 (1980).

18. Turk, Matthew and Alex Pentland. "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, pages 1-28 (September 1990).

Vita

Brian Singstock was born a poor, white child in a economically recessive small town just a few days shy of all Hallow's Eve in 1967. For his third birthday, his parents abandoned him. Luckily a pack of wolves took him in as one of their own. On his thirteenth birthday, the wolves returned him to the jungle...Northern Virginia. There he lived in the gutter, begging small coins in the dark hours of the night. He rebounded from such a setback with determination and naturally good looks and won an appointment to the Air Force Academy. While at the Academy, he was a member of the *Wings of Blue*, the infamous USAFA parachute team. In the Spring of 1990, he decided that the Academy had nothing else to offer him, and he left a Distinguished Graduate, with a B.S.E.E. From there it was on to Ohio to get his master's. While in Ohio he raised a yellow lab puppy that, one afternoon, found the same pack of wolves. Together they decided to travel the great north once again, assuming he graduated with his M.S.E.E. Well, needless to say, he graduated in the Winter of 1991, and off they went, the wolves, the man, and his dog. He now resides in Boston, Massachusetts, where he works at Hanscom AFB.

Permanent address: 138 Brown Avenue
Fairborn, Ohio 45324

December 1991 Master's Thesis

INFRARED TARGET RECOGNITION

Brian D. Singstock, Second Lieutenant, USAF

Air Force Institute of Technology, WPAFB OH 45433-6583

AFIT/GE/ENG/91D-49

Ed Zelnio
WRDC/AARA
Wright-Patt AFB OH 45433

In this thesis, three approaches were used for Automatic Target Recognition (ATR). These approaches were shape, moment and Fourier generated features, Karhunen-Loève Transform (KLT) generated features and Discrete Cosine Transform (DCT) generated features. The KLT approach was modelled after the face recognition research by Suarez, AFIT, and Turk and Pentland, MIT. A KLT is taken of a reduced covariance matrix, composed of all three classes of targets and the resulting 'eigenimages' are used to reconstruct the original images. The reconstruction coefficients for each original image are found by taking the dot product of the original image with each 'eigenimage'. These reconstruction coefficients were implemented as features into a three layer backprop with momentum network. Using the hold-one-out technique of testing data, the net could correctly differentiate the targets 100% of the time. Using standard features, the correct classification rate was 99.33%. The DCT was also taken of each image, and 16 low frequency Fourier components were kept as features. These recognition rates were compared to FFT results where each set contained the top five features, as determined by a saliency test. The results proved that the DCT and the FFT were equivalent concerning classification of targets.

Automatic Target Recognition, Target Classification, Infrared Imagery, Neural Nets 81

Unclassified Unclassified Unclassified UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.